HPC Performance Analysis of a Distributed Information Enterprise Simulation

James P. Hanna, Martin J. Walter, and Robert G. Hillman
Air Force Research Laboratory, Information Directorate (AFRL/IF), Advanced Computing Technology Branch, Rome, NY
{James.Hanna, robert.hillman}@rl.af.mil, Martin.Walter@afrl.af.mil

Abstract

Simulations of distributed information enterprises using the DIEMS framework[1] were performed on HPC clusters and SMP machines. The simulation results were analyzed with respect to the computation time involved in each process, the number of times processes were executed, the number of simulation rollbacks invoked during each simulation, simulation overhead, and the total wall clock time used to perform the simulation. The analysis identified several performance limitations and bottlenecks. One critical limitation addressed and eliminated was simultaneously mixing a periodic process model with an event driven model causing rollbacks. The second major factor limiting performance on cluster based systems was the cross-node communication. An optimization technique that exploits the knowledge of the publication and subscription paradigm of the information architecture being simulated was developed. This paper describes the simulation analysis, the modifications to the simulation models, the development of an optimization technique, and the impact of the code improvements on simulation performance.

1. Introduction

The Air Force is completing the development of a Distributed Information Enterprise Modeling and Simulation (DIEMS) framework under sponsorship of the High Performance Computer Modernization Program Common High Performance Computing Software Support Initiative (HPCMP/CHSSI). The DIEMS framework is a simulation system to analyze DoD deployable distributed information management systems. DIEMS establishes the necessary analysis capability allowing developers to identify and mitigate programmatic risk early within the development cycle to allow successful deployment of the associated systems[2,3,4].

The DIEMS simulator provides the framework, mechanism, and semantics to support the modeling and simulation of a wide variety of information enterprise architectures and implementations. The system defines topology, client functionality, and use-case scenarios that stimulate and drive the simulation separately through configuration files. One configuration file specifies the information enterprise topology and interconnectivity, as well as the compute resources and capacities of platforms that comprise the hardware infrastructure. A second file specifies the placement of individual JBI clients (SAs, PAs, LUSs, etc.) onto the hardware resources that make up the hardware infrastructure. Of course, individual hardware platforms may host any number of JBI clients. A third configuration file associates the information objects, duration of publication, rates of publication, etc. with specific JBI clients specified in the previous file. The DIEMS system reads these files, elaborates the specified enterprise architecture, instantiates clients on platforms, allocates tasks (publications and subscriptions) to specific clients, and executes the enterprise simulation. Simulation traces are recorded along with a variety of user-specified information and statistics. A number of analysis and display tools have also been developed to aid in the visualization and analysis of the simulation results. These tools can be used to assess the simulation as it progresses or after the fact to post process the results.

The development and simulation of a JBI notional architecture identifying infrastructural requirements was reported in[5]. This paper reports on the analysis of the performance impediments that were uncovered by the application of DIEMS to this atypical example of a distributed information enterprise. This atypical example was designed to stress the DIEMS framework and illuminate potential performance issues. We also discuss modifications made to the simulation framework to address these performance impediments and present results showing the performance improvements achieved.

2. Performance Analysis

Having successfully completed the functional and performance requirements for the DIEMS Alpha release, a concentrated performance optimization effort was executed. The optimization strategy was divided into three phases that were executed in sequential order, each leveraging the prior phase. The first phase was to profile the Alpha system and evaluate the individual functional blocks for performance. This phase included an investigation of the functional blocks that were consuming compute resources and an assessment of new algorithms or data structures to reduce the computational time. The second and third phases focused on reducing the rollbacks that were being generated within the SPEEDES parallel simulation framework. The second phase redesigned the platform processing algorithm to replace the inefficient time driven algorithm with a purely interrupt driven algorithm, reducing unnecessary processing and minimizing the effects of induced rollbacks. The third and final phase involved optimizing object placement on specific CPU nodes in order to reduce node to node interactions by clustering high traffic platforms on common CPU nodes.

3. Phase I

SPEEDES provides a built-in ability to instrument the simulation and analyze process performance. The SPEEDES instrumentation was used to profile the alpha system by recording total events and CPU time utilized for each functional block. The atypical, worst case test scenario was also generated to drive the simulation during this phase of the analysis. A significant aspect of this simulation configuration was that a network topology and scenario were established whereby communications were streamed continually between opposite ends of the enterprise. This scenario was atypical of a truly distributed enterprise but established an extremely difficult test case for the purpose of our analysis.

Figure 1 summarizes initial execution analysis that required over 20 days of CPU time with a wall clock of 5.3 days. Initial analysis indicated that over 3/4 of the CPU time was consumed within the SPEEDES framework which caused an initial concern relative to the architectural design used for DIEMS. This concern was later determined to be unfounded. We expected that the rollbacks were a significant contributor to the framework overhead. Another interesting observation was that almost 1/3 of the CPU time was consumed by a single event interface that simulated the platform-to-platform network routing.

Upon inspecting and analyzing the process routing algorithm it was determined that considerable time was being spent in evaluating the routing data structure. The algorithm was modified to include a caching function to eliminate the duplicative analysis. The overall result was a reduction in wall clock from 5.25 days to 8 hours. During the 8 hour run, the DIEMS core consumes 5.3 CPU hours with the SPEEDES core consuming 203 CPU hours. The code was next analyzed with respect to evaluating the event interfaces relative to rollback and the associated processing time.

Figure 2 illustrates the rollback related process times with four interfaces being the major contributors; "NetRouterReciever", "ExecuteRouting", "ProcessRouting", and "Processing". The first three functional interfaces establish the network interface of the platforms. The behavior emulates a switching router that facilitates a store and forward mechanism. The analysis identified that the store and forward mechanism was responsible for the rollbacks associated with these event interfaces. The network interface was modified combining the functions into a single interface that emulates a router performing a simple receive and forward function.

Figure 3 illustrates the relative processing times of the top CPU intensive event interfaces at the completion of this first code optimization phase. The event execution times are now somewhat balanced with "NetRouterReciever" and "Processing" consuming 60% of the time, a reasonable state since they reflect the main functions, processing atomic behavior and routing the messages. At the completion of this phase the overall wall clock processing time was reduced from 5.25 days (126 hours) to 3.6 hours with rollbacks reduced from 91 million to 43 million.

4. Phase II

The second optimization phase focused on redesigning the platform processing algorithm to reduce execution time and rollbacks. In this discussion, the term platform is used to identify one of the virtual computer objects on the simulated network/enterprise. The entire simulation is comprised of interconnected platforms and network domains (or routers) given some specific enterprise topology.

The initial platform processing algorithm evaluated the task queue on each platform based on an interval timeout to schedule platform processing events. At each interval all events expected to occur before the next interval timeout were scheduled via the SPEEDES API call SCHEDULE_*SomeEvent*(). This API call enters the given event into the SPEEDES event queue for processing. On multiprocessor systems each node upon which the distributed simulation runs processes into the future independently of other processor nodes. When events that occur in the past are received by a processing node, the simulation must roll back. To achieve this, all events at the receiving node must be undone; the simulator on that node must reset time to accommodate the past event and then reprocess all of the events that were rolled back.

The judicial selection of an appropriate timeout interval is critical to minimizing the number of rollbacks the system must process. Rollbacks are very expensive computationally. Unfortunately, selecting the correct interval depends upon the granularity of the network traffic scenarios and the actual node placement of objects within the simulation. In the general case, no appropriate selection can be made.

To alleviate the problems associated with the interval timeout and to minimize simulator rollbacks an event driven approach to platform processing was devised. This new approach, albeit more complicated, has the advantage of assuring that at most a single event per platform will never require reprocessing when a rollback occurs. The event-based algorithm evaluates the platform's task queue and schedules the current event for the given platform. Next it evaluates the task queue to determine when the next event occurs and schedules a wakeup event on itself. This process continues until there are no more events to schedule. This redesign of the platform processing algorithm reduced the number of rollbacks from 43 million to 22 million and decreased the wall clock processing time from 3.6 hours to 1.8 hours.

5. Phase III

The final optimization phase focused on object placement and required the development of a more realistic operational scenario where information clusters existed between geographical local communities.

The DIEMS simulation objects, specified in the topology configuration file, are distributed across the HPC nodes using a decomposition algorithm. Platforms are distributed with their respective domains, and by default, domains are distributed in a block method. The block method places the first ***n*** processes on the first HPC node, the next ***n*** processes on the next node, etc.

The message traffic between HPC nodes is strongly dependent upon which domains and platforms are on which HPC nodes and is also strongly dependent upon the details of the message traffic specified in the scenario configuration file. It is an oversimplification to assume that domains near each other in the topology numbering scheme will necessarily dominate the communications. In order to remove the uncontrolled character of the internode's communication, a scheme was developed to place domains that share the highest number of inter-domain messages on the same CPU. This controls and reduces the inter-CPU communication during the simulation and reduces the number of simulation rollbacks processed.

In order to place domains on the CPUs based on the message traffic, the number of messages between domains must be known. One method to obtain the inter-domain message traffic before executing the simulation is to enumerate the messages. For deterministic protocols such as Jini PUB/SUB, all messages can be known apriori, by examining the configuration files.

Enumerating the messages has the additional advantage of enabling verification of the simulation, in that the simulation should produce the same messages, but distributed in time.

Enumerating the messages is based on knowledge of the publication and subscription paradigm being used. A message enumerator for the Jini PUB/SUB architecture was developed to provide a summary of the number of messages. The result is a list of the messages identified by the message Class, Detail and Data Type, received by each platform in the simulation. It also generates a summary list of the inter-domain message traffic, which can be represented by a set of nodes and edges, where the nodes are the domain IDs, and the "edges" are the inter-domain connections. The inter-domain message traffic list is ordered by the decreasing number of messages on each edge, as can be seen in Figure 4.

Placement of the domains and their associated platforms is performed by using the order that the domains appear in the list of "edges". Figure 4 shows an idealized multiprocessor with five CPUs, the topology configuration for the domains, and a sample "edge" file with domain IDs and the numbers of messages exchanged between the domains. The decomposition is accomplished by checking each domain on an "edge" for previous placement. If one domain on the "edge" has been placed, the other is placed on the same CPU. If neither has been placed, both domains are placed on the same CPU. After all of the domains have been placed on the CPUs an Object Placement file is written. DIEMS was modified to optionally read this file, and distribute the domains and their platforms accordingly during simulation initialization.

A variation on the Atypical example configuration used in Hanna, et al.[1] was employed to evaluate the simulation object placement with the DIEMS framework. This configuration consists of 67 domains and 152 platforms. The communication architecture employed is JiniPub/Sub. The domain connectivity is seen in Figure 4 and Figure 5, where the clusters are indicated by the ellipses, the domains are indicated by the numbered circles, and the inter-domain connectivity is indicated by the black lines. The circles not contained in any ellipse indicate a fifth cluster. Verification of the configuration and its' implementation in the DIEMS framework was accomplished by comparing the enumerated number of messages for each type, class and detail with the number obtained from the simulation output. More than 1.5 million messages were received during the simulation. This configuration has a strong grouping of the message traffic into five clusters, with about 3% of the communication between the clusters.

The configuration was simulated on an HPC cluster using differing numbers of CPUs. Figure 6 shows the time to completion for the simulations as the number of CPUs was varied. The figure shows two data sets; one for the default block mode object decomposition, and one for the message-based decomposition (BACH). As the number of CPUs is increased, the use of message-based decomposition results in improved time to completion as compared to block mode. Further, the use of block mode shows the usual upturn in completion time as the number of processors increases. This upturn is not apparent with the use of message-based decomposition.

The decomposition algorithm uses only the order and connections of the inter-domain message traffic. The degree of the clustering of the message traffic effects the placement, and the simulation performance. For cases where the inter-cluster message traffic is a larger percentage of the total traffic, the inter-cluster edges are placed earlier and the clusters may not be cleanly distributed among the CPUs.

Consider the case of a cluster of seven domains, domains 42 through 48 in Figure 5. For a simulation where the message traffic between domains 48 and 58 is greater than the traffic between 48 and any other domain in that cluster, domains 48 and 58 are placed on the same CPU, to reduce the number of cross CPU messages. The result is that those two clusters will reside on the same CPU. Even for modest percentages of inter-cluster message traffic, this can occur since the percentage is inversely proportional to the sum of the traffic among the cluster's domains, while the placement is based on a comparison of individual "edges."

6. Summary

The Distributed Information Enterprise Modeling and Simulation (DIEMS) framework under sponsorship of the High Performance Computer Modernization Program Common High Performance Computing Software Support Initiative (HPCMP/CHSSI) and its performance enhancements have been described. This paper reported on the analysis of performance impediments that were uncovered in the application of DIEMS to an atypical worst case example of a distributed information enterprise. These limitations included excessive CPU time in ProcessRouting, extremely large numbers of simulation rollbacks, and uncontrolled inter-process communications, which aggravated the rollback problems.

These issues were addressed by implementing a caching algorithm for the ProcessRouting; developing an internal scheduler, which eliminates group rollbacks and reduces the processing time on unnecessary updates; and by implementing an inter-process communication based simulation object placement algorithm.

These modifications reduced simulation run time for eight CPUs from more than five days to less than 25 minutes, reducing the simulation wall clock to 0.3% of its original value.

References

1. Hanna, James P., Robert G. Hillman, and Martin J. Walter, "Modeling and Simulation of the Joint Battlespace Infosphere Scalability." *Proceedings of the Summer Computer Simulation Conference,* July 2001.

2. Hanna, James P., Robert G. Hillman, and Martin J. Walter, "Modeling and Simulation of the Joint Battlespace Infosphere Scalability." *Proceedings of Summer Computer Simulation Conference*, July 2001.

3. Hanna, James P. and Robert G. Hillman, "SPEEDES for Distributed Information Enterprise Modeling." *Proceedings of the International Society for Optical Engineering Enabling Technologies for Simulation Science,* April 2002.

4. Hanna, James P., Robert G. Hillman, and Martin J. Walter, "Modeling the Joint Battlespace Infosphere."*Proceedings of the 6th International Command and Control Research and Technology Symposium*, June 2001.

5. Hanna, James P., Martin J. Walter, Robert G. Hillman, and Lois D. Walsh, PhD., "Analysis of a JBI Pub/Sub Architecture's Infrastructure Requirements.", *Proceedings of the International Society for Optical Engineering Enabling Technologies for Simulation Science*, April 2003.

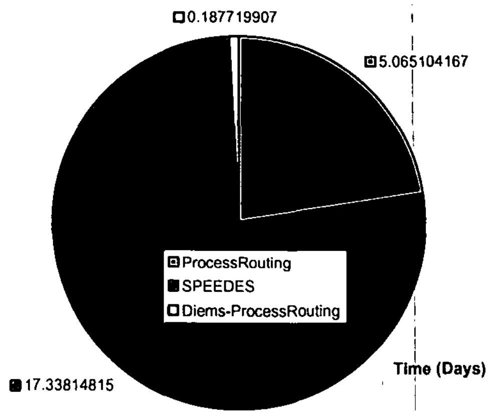


Figure 1. Total processing time

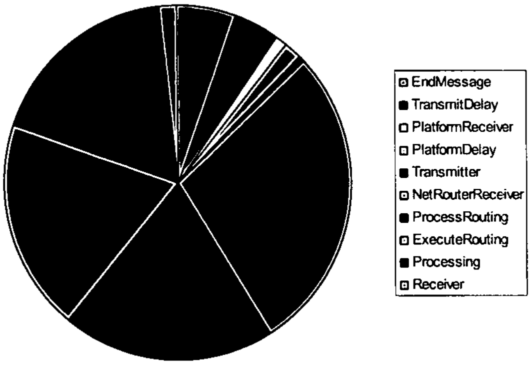


Figure 2. Rollback relative processing times

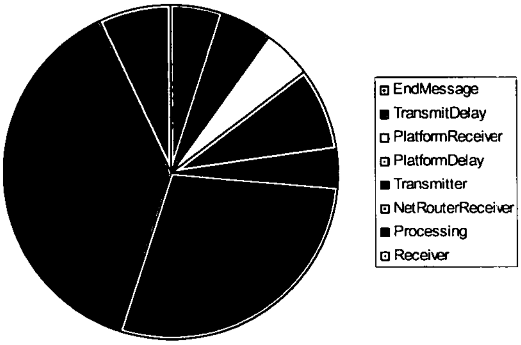
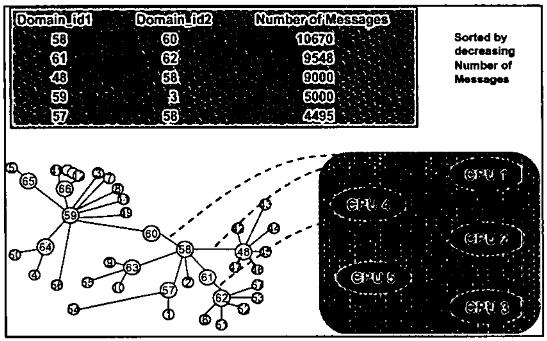


Figure 3. Final relative processing times



Domain_id1	Domain_id2	Number of Messages
58	60	10670
61	62	9548
48	58	9000
59	3	5000
57	58	4495

Sorted by decreasing Number of Messages

Figure 4. Simulation object placement scheme

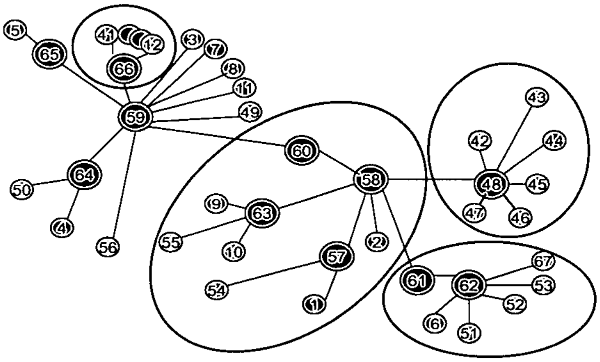


Figure 5. Communication cluster

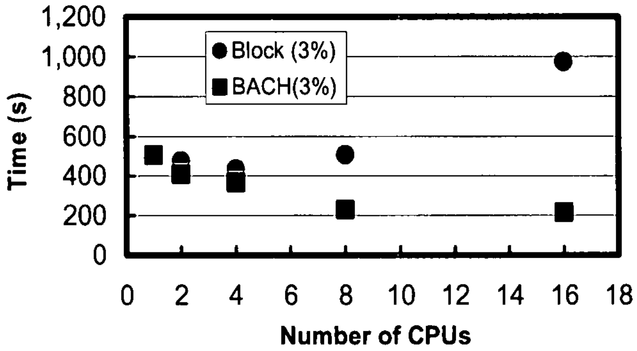


Figure 6. Simulation time vs. object decomposition algorithms